

Part II: Interfacing Mechanism Model to CLARAty Abstractions

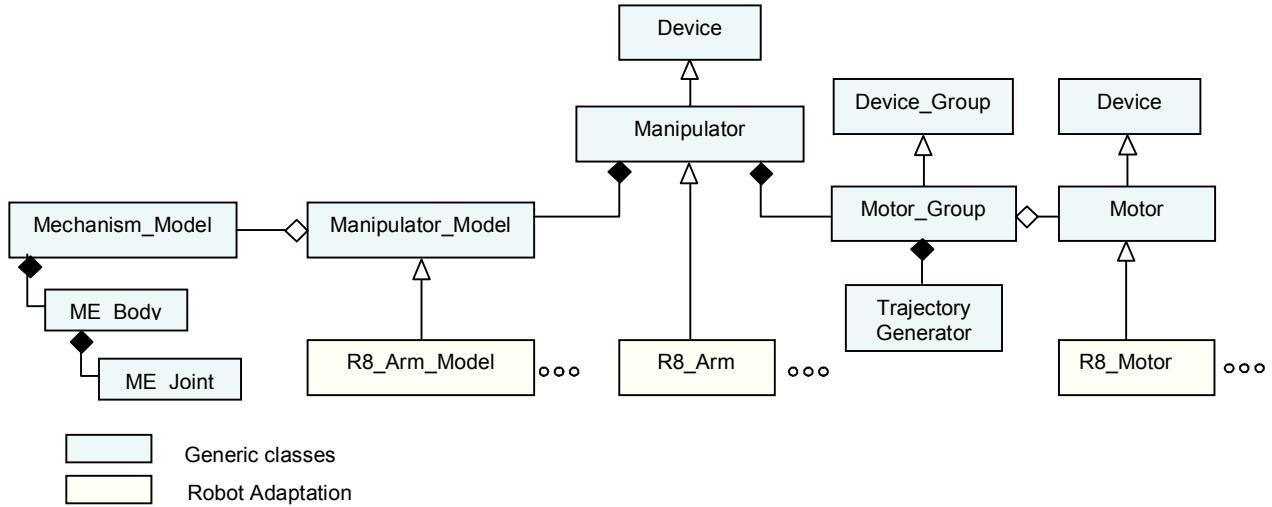


Figure 9: Use of [Manipulator_Model](#) class

The second part of this document describes how this mechanism model shall be interfaced with the CLARAty locomotion and manipulation abstractions.

11. Software Interfaces

The class hierarchy for the [Mechanism_Model](#) software package is illustrated in Figure 8. Figure 9 shows an example using the manipulator classes to illustrate the interface of the mechanism model to the control classes (generic physical components). The [Mechanism_Model](#) classes shall be used either as a stand-alone package for kinematic analysis or as part of the control software for the robotic system.

- 11.1. [Mechanism_Model](#) may be used directly by the robot control system or through interface classes that present simpler and restricted interfaces for common types of mechanisms. For example, the interface class for manipulators shall be the [Manipulator_Model](#) class. Similarly, for wheeled mobile robots, it shall be [Wheeled_Locomotor_Model](#) and, for walking mobile robots, it shall be [Legged_Locomotor_Model](#).
- 11.2. The interface classes and their corresponding control classes shall enable the mechanical sub-systems to be controlled independently. This is the typical mode used for robotic operations. For example, a rover with its mounted arm can be treated as two independent control systems by utilizing the [Manipulator_Model](#) and [Wheeled_Locomotor_Model](#) interface classes for the arm and the rover respectively. Alternatively, one can use the [Mechanism_Model](#) instead to simultaneously coordinate the arm with the rover motions.
- 11.3. Adaptations of [Manipulator_Model](#), [Wheeled_Locomotor_Model](#) and [Legged_Locomotor_Model](#) can override generic kinematics algorithm implementations with specialized algorithms whenever necessary.

- 11.4. Specialized kinematic algorithm implementations shall use parameter information from `Mechanism_Model` classes.
- 11.5. `Manipulator_Model`, `Wheeled_Locomotor_Model` and `Legged_Locomotor_Model` classes shall support serialization of their model information. This will support both writing these models to file storage and retrieving and instantiating the models from their respective files(see Section 6).
- 11.6.

12. Model Classes for Manipulators and Locomotors

This section addresses models for generic manipulators, wheeled locomotors, and legged locomotors. In the requirements below, we will use the `Manipulator_Model` class as an example. Figure 9 illustrates the relationships among `Manipulator_Model` interface class, `Mechanism_Model` class, and `Manipulator` control classes. The same applies for wheeled, legged and hybrid locomotors.

- 12.1. `Manipulator_Model` shall reference the `Mechanism_Model` class for accessing the kinematic model data on the manipulator.
- 12.2. Adaptations of the `Manipulator_Model` class may define specialized implementations (e.g., closed-form solution) of forward, inverse, and differential kinematics to override, as necessary, generic solutions available from the `Mechanism_Model`. Adaptations of the `Wheeled_Locomotor_Model` class shall define specialized implementations of flat-ground open-loop driving and slope differential driving kinematics algorithms. Adaptations of the `Legged_Locomotor_Model` class shall define specialized implementations of flat-ground open-loop walking and slope differential walking kinematics algorithms.
- 12.3. `Manipulator_Model` subclasses shall be concrete classes. Examples of such classes include `R8_Arm_Model` and `FD_Mast_Model`.

13. Control Classes for Manipulators and Locomotors

This section applies to the generic manipulator, wheeled locomotor, and legged locomotor classes. The requirements below are only intended to describe the relationship between the manipulator/locomotor classes and their corresponding model classes. The requirements below use the `Manipulator` class as an example. Figure 9 illustrates the `Manipulator` and the associated control classes. The same applies for wheeled, legged and hybrid locomotors.

- 13.1. `Manipulator` class shall inherit from generic `Device` class and contain the following:
 - `Manipulator_Model`
 - `Motor_Group`
- 13.2. `Motor_Group` class shall define the correspondence between manipulator joints and actual motors.
- 13.3. `Manipulator` class shall bind the `ME_Joint` objects to the individual motors
- 13.4. `Manipulator` class shall have APIs that support various motion control modes for:
 - 13.4.1. Individual joint/wheel control
 - 13.4.2. Coordinated joint/wheel control

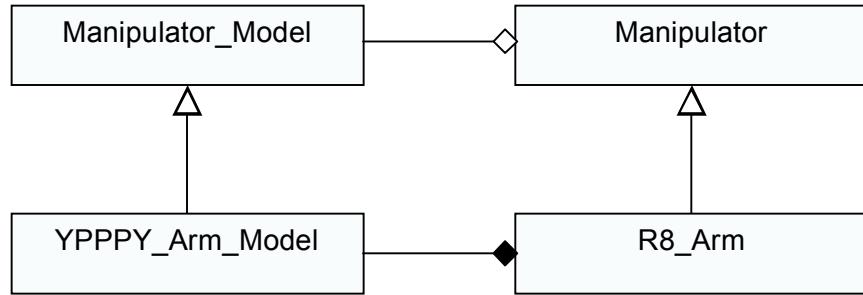


Figure 10: Inheriting specific models from [Manipulator_Model](#)

- 13.4.3. Following a prescribed path or trajectory (e.g. straight line motion of the end effector of a manipulator)
- 13.4.4. Gravity compensation for manipulators
- 13.4.5. Sensor-based control such as force control, compliance control, visual servoing, and so on.
- 13.5. [Motor_Group](#) may use a [Trajectory Generator](#) object for closed loop coordination of multiple joints. The [Manipulator](#) class shall generate set points using generic or specialized kinematic algorithms and feed them to the [Trajectory Generator](#) through the [Motor_Group](#) API.
- 13.6. [Manipulator](#) subclasses shall be concrete classes. Examples include [R8_Arm](#) and [FD_Mast](#).

14. Examples of Model Instantiation

```

// Create a mechanism model of the K-9 arm
YPPP_Y_Arm_Model ypppy_model("k9_arm_model.xml");

// Create a R7_Arm manipulator with a YPPP_Y model
R7_Arm           r7_arm(ypppy_model);
  
```

An example to create a mechanism model of a walking robot shall be as follows

To create an arm that uses a yaw-pitch-pitch-yaw (YPPP_Y) configuration, one can create a YPPP_Y kinematic model which is a specialization of [Mechanism_Model](#).

```

// Create a mechanism model of the rover body
Mechanism_Model   rover_model("body_model.xml");

// Attach a leg to the "leg1_mnt" frame defined in "body_model.xml"
// using the leg model in "leg_model.xml"
rover_model.attach(("leg_model.xml", "leg1_mnt"));

// Attach a leg to the "leg2_mnt" frame defined in "body_model.xml"
// using the leg model in "leg_model.xml"
rover_model.attach(("leg_model.xml", "leg2_mnt"));
rover_model.save("entire_model.xml");
  
```

Appendix A: Document Definition Example

This appendix contains two examples of the mechanism model XML files, and their corresponding Document Type Definition for mechanism model. Below is the simple example:

```

<?xml version="1.0" encoding="US-ASCII" ?>
<!DOCTYPE Mechanism_Model>
- <!--
*=====
*=                               /-/ CLARAty /-/
*=====
* @file simple_sample1.xml
*
* An example of the use of the DTD
*
* Designed by:      Diaz-Calderon
* @author:          Diaz-Calderon
* @date:            June 15, 2004
*
* Software Use Notice
* -----
* http://claraty.jpl.nasa.gov/sw_use_notice.html or
* ./share/sw_use_notice.txt
*
* (C) 2004, Jet Propulsion Laboratory, California Institute of Technology
*
* $Revision:$
*-----
-->
<Mechanism_Model name="simple_two_link" version="1.0">
- <ME_Body name="link1">
-   <ME_Joint name="joint1" type="revolute" actuated="true" offset="0"></ME_Joint>
-   <Frame name="ref1" type="reference">
-     <Position x="2.0" y="0" z="0" />
-     <Rotation type="Euler_ZYZ" r1="M_PI/2" r2="0" r3="0" />
-   </Frame>
- </ME_Body>

- <ME_Body name="link2" parent="link1">
-   <ME_Joint name="joint2" type="revolute" actuated="true" offset="0"></ME_Joint>
-   <Frame name="ref2" type="reference">
-     <Position x="10" y="0" z="0" />
-     <Rotation type="Euler_XYZ" r1="M_PI/2" r2="M_PI" r3="0" />
-   </Frame>
- </ME_Body>

</Mechanism_Model>
- <!--
EOF simple_sample1.xml
-->

```

Below is a second more complete sample model for a three link manipulator.

```

<?xml version="1.0" encoding="US-ASCII" ?>
<!DOCTYPE Mechanism_Model (View Source for full doctype...)>
- <!--
*= ====== CLARAty /-/
* @file simple_sample1.xml
*
* An example of the use of the DTD
*
* Designed by: Diaz-Calderon
* @author: Diaz-Calderon
* @date: June 15, 2004
*
* Software Use Notice
* -----
* http://claraty.jpl.nasa.gov/sw_use_notice.html or
* ../share/sw_use_notice.txt
*
* (C) 2004, Jet Propulsion Laboratory, California Institute of Technology
*
* $Revision:$
*-----
-->
- <Mechanism_Model name="two_link" version="1.0">
- <ME_Body name="link1">
-   <ME_Joint name="joint1" home="1.0" type="revolute" actuated="true" offset="0">
-     <Joint_Limits min="-M_PI" max="M_PI" vmax="0.25" torque_min="0"
-       torque_max="0" />
-       <Joint_Stiffness kx="0.1" ky="0.2" kz="0.3" />
-   </ME_Joint>
-   <Frame name="ref1" type="reference">
-     <Position x="0" y="0" z="0" />
-     <Quaternion qi="0" qj="0" qk="0" qs="1" />
-   </Frame>
-   <Mass_Properties mass="0.25">
-     <CM>
-       <Position x="1" y="1" z="1" />
-       <Quaternion qi="0" qj="0" qk="0" qs="1" />
-     </CM>
-     <Inertia xx="0" yy="0" zz="0" />
-   </Mass_Properties>
-   <Bounding_Shape>
-     <Obj_Parms name="obj20" type="box" link="IDD_COL_EL_JOINT"
-       is_container="true" level="1" is_aligned="no" ignore_collisions="false" tol="-
-       1.0">
-       <Position x="0" y="0" z="0" />
-       <Quaternion qi="0" qj="0" qk="0" qs="1" />

```

```

        <Dimension dim-x="0" dim-y="0" dim-z="0" />
    </Obj_Parms>
    -<Obj_Parms name="obj21" type="box" parent="obj20"
      link="IDD_COL_LINK_AUTO" is_container="none" level="2" is_aligned="no"
      ignore_collisions="false" tol="-1.0">
        <Position x="0" y="0" z="0" />
        <Quaternion qi="0" qj="0" qk="0" qs="1" />
        <Dimension dim-x="0" dim-y="0" dim-z="0" />
    </Obj_Parms>
    -<Obj_Parms name="obj22" type="box" parent="obj20"
      link="IDD_COL_LINK_AUTO" is_container="none" level="1" is_aligned="no"
      ignore_collisions="false" tol="-1.0">
        <Position x="0" y="0" z="0" />
        <Quaternion qi="0" qj="0" qk="0" qs="1" />
        <Dimension dim-x="0" dim-y="0" dim-z="0" />
    </Obj_Parms>
</Bounding_Shape>
<Display_Graphics path="graphics_file.vrml" />
</ME_Body>
-<ME_Body name="link3">
  -<ME_Joint name="joint3" type="revolute" actuated="true" offset="0" home="0">
    <Joint_Limits min="-M_PI" max="M_PI" vmax="1.0" torque_min="0"
      torque_max="0" />
  </ME_Joint>
  -<Frame name="ref3" type="local">
    <Position x="1" z="10" y="0" />
    <Quaternion qi="0" qj="0" qk="0" qs="1" />
  </Frame>
  -<Mass_Properties>
    -<CM>
      <Position x="0" y="0" z="0" />
      <Quaternion qi="0" qj="0" qk="0" qs="1" />
    </CM>
    <Inertia xx="1" yy="2" zz="3" />
  </Mass_Properties>
</ME_Body>
-<ME_Body name="link2" parent="link1">
  -<ME_Joint name="joint2" type="revolute" actuated="false" offset="M_PI_2"
    home="0">
    <Joint_Limits min="-M_PI" max="M_PI" vmax="1.0" torque_min="-1.0"
      torque_max="1.0" />
    <Joint_Stiffness kx="0.1" ky="0.2" kz="0.3" />
    <Joint_Constraint expr="2*joint1" />
  </ME_Joint>
  -<Frame name="ref2" type="reference">
    <Position x="10" y="0" z="0" />
    <Quaternion qi="0" qj="0" qk="0" qs="1" />
  </Frame>
  -<Frame name="sens1" type="local">
    <Position x="4" y="10" z="0" />
  </Frame>

```

```
<Quaternion qi="0" qj="0" qk="0" qs="1" />
</Frame>
- <Frame name="force1" type="local">
    <Position x="3" y="2" z="0" />
    <Quaternion qi="0" qj="0" qk="0" qs="1" />
</Frame>
- <Mass_Properties mass="0.25">
    <CM>
        <Position x="1" y="1" z="1" />
        <Quaternion qi="0" qj="0" qk="0" qs="1" />
    </CM>
    <Inertia xx="0" yy="0" zz="0" />
</Mass_Properties>
- <Bounding_Shape>
- <Obj_Parms name="obj30" type="box" link="IDD_COL_EL_JOINT"
is_container="true" level="1" is_aligned="no" ignore_collisions="false" tol="-1.0">
    <Position x="0" y="0" z="0" />
    <Quaternion qi="0" qj="0" qk="0" qs="1" />
    <Dimension dim-x="0" dim-y="0" dim-z="0" />
</Obj_Parms>
</Bounding_Shape>
<Display_Graphics path="graphics_file.vrml" />
</ME_Body>
</Mechanism_Model>
- <!--
EOF sample1.xml
-->
```

The document data type is shown below:

```

<?xml version="1.0" encoding="US-ASCII"?>

<!--
=====
*=                               /-/ CLARAty /-/
=====

* @file sample1.xml
*
* An example of the use of the DTD
*
* Target OS: Generic (VxWorks/UNIX/Linux)
*
* Designed by:      Diaz-Calderon
* @author:          Diaz-Calderon
* @date:           June 15, 2004
*
* Software Use Notice
* -----
* http://claraty.jpl.nasa.gov/sw_use_notice.html or
* ../share/sw_use_notice.txt
*
* (C) 2004, Jet Propulsion Laboratory, California Institute of Technology
*
* $Revision:$
*
* Modification History
* -----
* 002,18jun04,adc Implemented changes per meeting with Issa
* 001,15jun04,adc updated to conform with the requirements document
* 000,27apr04,adc original writing of the dtd
*-----
-->

<!----- M E C H A N I S M _ M O D E L ----->
<!ELEMENT Mechanism_Model ((ME_Body+)
                           | (Block_DH, Block_Joint))>
<!ATTLIST Mechanism_Model
  name    ID                      #REQUIRED
  version CDATA                  #REQUIRED>

<!----- M E _ B O D Y ----->
<!-- me-body requires at least one frame that describes the fixed -->
<!-- transform from parent reference frame to the body reference -->
<!-- frame. The required node will define the reference frame of the -->
<!-- body relative to the reference frame of the parent; i.e., the -->
<!-- body-to-parent transform. All other nodes (including mount nodes -->
<!-- must be defined relative to the body reference frame. -->
<!ELEMENT ME_Body (ME_Joint,
                   Frame+,
                   Mass_Properties?,
                   Bounding_Shape?,
                   Display_Graphics?)>
<!ATTLIST ME_Body
  name    ID                      #REQUIRED
  parent  IDREF                  #IMPLIED>

<!----- M E _ J O I N T ----->
<!ELEMENT ME_Joint (Joint_Limits?, Joint_Stiffness?, Joint_Constraint?)>
<!ATTLIST ME_Joint

```

```

name      ID                      #REQUIRED
type      (revolute | prismatic) "revolute"
actuated  (true | false) "true"
offset    NMOKEN "0"
home     NMOKEN "0">>

<!-- J O I N T _ L I M I T S -->
<!ELEMENT Joint_Limits EMPTY>
<!ATTLIST Joint_Limits
  min      NMOKEN "-M_PI"
  max      NMOKEN "M_PI"
  vmax    NMOKEN "1.0"
  torque_min NMOKEN "0"
  torque_max NMOKEN "0">>

<!-- J O I N T _ S T I F F N E S S -->
<!ELEMENT Joint_Stiffness EMPTY>
<!ATTLIST Joint_Stiffness
  kx NMOKEN                      #REQUIRED
  ky NMOKEN                      #REQUIRED
  kz NMOKEN                      #REQUIRED>

<!-- J O I N T _ C O N S T R A I N T -->
<!ELEMENT Joint_Constraint EMPTY>
<!ATTLIST Joint_Constraint
  expr CDATA                      #REQUIRED>

<!-- M A S S - P R O P E R T I E S -->
<!ELEMENT Mass_Properties (CM, Inertia)>

<!-- If the values for rho and volume are given, mass it is a derived -->
<!-- parameter mass = rho*volume -->
<!ATTLIST Mass_Properties
  rho    NMOKEN                  #IMPLIED
  volume NMOKEN                  #IMPLIED
  mass   NMOKEN                  #IMPLIED
>

<!-- I N E R T I A -->
<!ELEMENT Inertia EMPTY>

<!-- Principal moments of inertia of the me-body. This assumes that -->
<!-- the frame located at the center of mass represent the principal -->
<!-- axes of the body. -->
<!ATTLIST Inertia
  xx NMOKEN                      #REQUIRED
  yy NMOKEN                      #REQUIRED
  zz NMOKEN                      #REQUIRED>

<!-- B O U N D I N G _ S H A P E -->
<!ELEMENT Bounding_Shape (Obj_Parms)+>

<!ELEMENT Obj_Parms (Position, Quaternion, Dimension)>
<!ATTLIST Obj_Parms
  name      ID                      #REQUIRED
  parent    IDREF                  #IMPLIED

```

```

type          (box | cyl | no_shape)      #REQUIRED
is_container  (true | false | none)       #REQUIRED
level         (1 | 2 | 3)                  #REQUIRED
link          NMOKEN                     #REQUIRED
is_aligned    (yes | no)                 #REQUIRED
ignore_collisions CDATA                 #REQUIRED
tol           NMOKEN                     #REQUIRED>

<!------- D I S P L A Y _ G R A P H I C S ----->
<!ELEMENT Display_Graphics EMPTY>
<!ATTLIST Display_Graphics
  path CDATA                                #REQUIRED>

<!------- D H _ P A R A M E T E R ----->
<!-- Denavit-Hartenberg parameters which can use either Craig's or -->
<!-- Paul's convention -->
<!ELEMENT DH_Parameter (DHC | DHP)>
<!ATTLIST DH_Parameter
  name   ID                      #REQUIRED
  length NMOKEN                #REQUIRED
  twist   NMOKEN                #REQUIRED
  offset NMOKEN "0"
  angle   NMOKEN "0">

<!------- H O M O G E N E O U S   T R A N S F O R M ----->
<!-- Can use a number of Euler angle conventions -->
<!ELEMENT HTrans (Position, Rotation) >

<!------- F R A M E ----->
<!-- a Frame represents a ref-frame, actuator-frame and -->
<!-- sensor-frame. There can only be one and only one ref-frame for a -->
<!-- given body. It is an error to define more than one reference -->
<!-- frames in a body. -->
<!ELEMENT Frame (DH_Parameter | HTrans | (Position, Quaternion))>
<!ATTLIST Frame
  name   ID                      #REQUIRED
  type   (reference | local) "local">

<!------- C M ----->
<!-- coordinates of the center of mass relative to the body reference -->
<!-- frame. The orientation of the principal axes relative to the -->
<!-- reference frame is given by the rotation matrix in the -->
<!-- transform. -->
<!ELEMENT CM (Position, Quaternion)>

<!------- R O T A T I O N ----->
<!ELEMENT Rotation EMPTY>
<!ATTLIST Rotation
  type  (EULER_ZYZ | EULER_XYZ | EULER_ZYX )
  r1   NMOKEN "0"
  r2   NMOKEN "0"
  r3   NMOKEN "0">

<!------- Q U A T E R N I O N ----->
<!ELEMENT Quaternion EMPTY>
<!ATTLIST Quaternion
  qi   NMOKEN "0"

```

```
qj NMTOKEN "0"
qk NMTOKEN "0"
qs NMTOKEN "1">>

<!----- P O S I T I O N ----->
<!ELEMENT Position EMPTY>
<!ATTLIST Position
  x NMTOKEN "0"
  y NMTOKEN "0"
  z NMTOKEN "0">

<!----- D I M E N S I O N ----->
<!ELEMENT Dimension EMPTY>
<!ATTLIST Dimension
  dim-x NMTOKEN "0"
  dim-y NMTOKEN "0"
  dim-z NMTOKEN "0">

<!----- B L O C K - D H ----->
<!ELEMENT Block_DH (DH_Parameter+)>

<!----- B L O C K - J O I N T ----->
<!ELEMENT Block_Joint (ME_Joint+)>

<!-- EOF mechanism_model.dtd -->
```

Appendix B: Input Representations

This appendix summarizes three input representations: (a) Craig's D-H, (b) Paul's D-H, and (c) Zero Position parameters

Craig's definition of D-H parameters

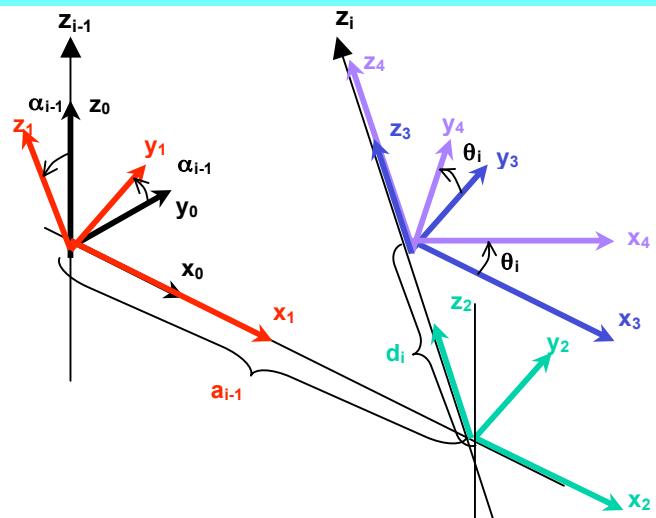
T1 = twist α_{i-1} about x_0 axis

T2 = length a_{i-1} along x_1 axis

T3 = offset d_i along z_2 axis

T4 = rotate θ_i about z_3 axis

$$T_i^{i-1} = \begin{pmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_i & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$



Paul's definition of D-H parameters

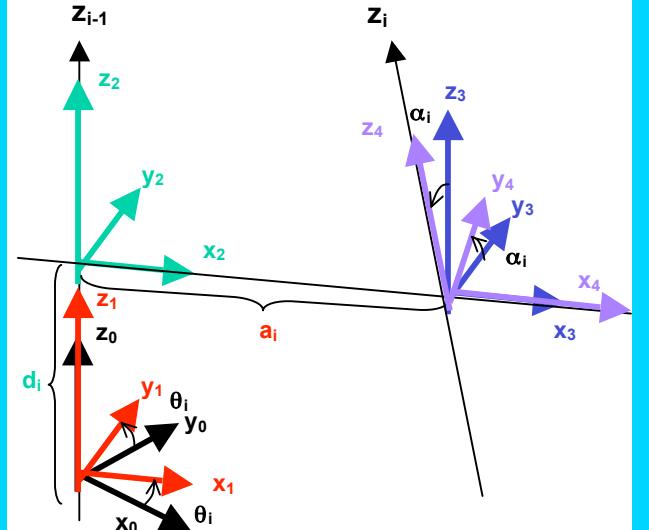
T1 = rotate θ_i about z_0 axis

T2 = offset d_i along z_1 axis

T3 = length a_i along x_2 axis

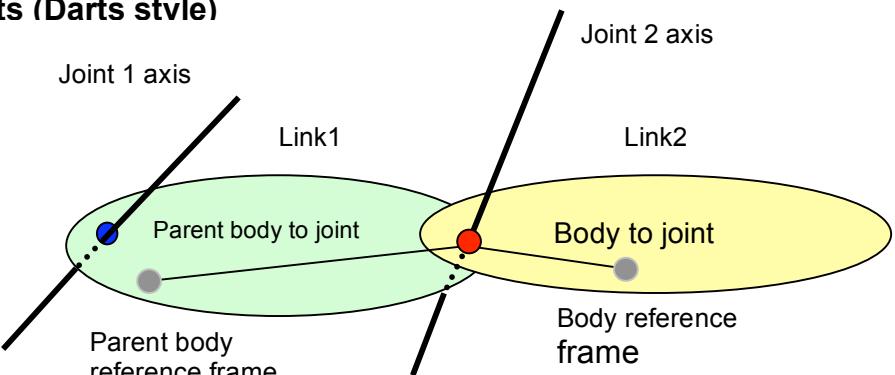
T4 = twist α_i about x_3 axis

$$T_i^{i-1} = \begin{pmatrix} c\theta_i & -s\theta_i c\alpha_i & -s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & s\theta_i c\alpha_i & -s\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Zero-Position Data Inputs (Darts style)

Define parameters at the zero configuration.



Body to joint = xb2, yb2, zb2 (default = 0 0 0)
 Parent body to joint = xp2, yp2, zp2 (default = 0 0 0)
 Joint axis = ax2, ay2, az2

Body name = Link2
 Parent body name = Link1
 Joint type = rotational

Mass; Inertia; CM inertia
 Body to CM
 Flexible body deformation parameters
 (nodal vector, stiffness, damping)